Cloud Search Service

Quick Start

Issue   01
Date   2023-03-28

HUAWEI TECHNOLOGIES CO., LTD.

Contents

1 Getting Started with Elasticsearch

This section describes how to use Elasticsearch for product search. You can use the Elasticsearch search engine of CSS to search for data based on the scenario example. The basic operation process is as follows:

- **Step 1: Create a Cluster**
- **Step 2: Import Data**
- **Step 3: Search for Data**
- **Step 4: Delete the Cluster**

Scenario Description

A women's clothing brand builds an e-commerce website. It uses traditional databases to provide a product search function for users. However, due to an increase in the number of users and business growth, the traditional databases have slow response and low accuracy. To improve user experience and user retention, the e-commerce website plans to use Elasticsearch to provide the product search function for users.

This section describes how to use Elasticsearch to provide the search function for users.

Assume that the e-commerce website provides the following data:

```
{
"products":[
{"productName":"Latest art shirts for women in autumn 2017","size":"L"}
{"productName":"Latest art shirts for women in autumn 2017","size":"M"}
{"productName":"Latest art shirts for women in autumn 2017","size":"S"}
{"productName":"Latest jeans for women in spring 2018","size":"M"}
{"productName":"Latest jeans for women in spring 2018","size":"S"}
{"productName":"Latest jeans for women in spring 2017","size":"L"}
{"productName":"Latest casual pants for women in spring 2017","size":"S"}
]
}
```

Step 1: Create a Cluster

Create a cluster using Elasticsearch as the search engine. In this example, suppose that you create a cluster named **Sample-ESCluster**. This cluster is used only for getting started with Elasticsearch. For this cluster, you are advised to select **ess.spec-4u8g** for **Node Specifications**, **High I/O** for **Node Storage Type**, and **40**

GB for **Node Storage Capacity**. For details, see **Creating an Elasticsearch Cluster in Non-Security Mode**.

Create a cluster using Elasticsearch as the search engine. In this example, suppose that you create a cluster named **Sample-ESCluster**. This cluster is used only for getting started with Elasticsearch. For this cluster, you are advised to select **ess.spec-4u8g** for **Node Specifications**, **High I/O** for **Node Storage Type**, and **40 GB** for **Node Storage Capacity**. For details, see **Creating an Elasticsearch Cluster in Security Mode** or **Creating an Elasticsearch Cluster in Non-Security Mode**.

After you create the cluster, switch to the cluster list to view the created cluster. If the **Status** of the cluster is **Available**, the cluster is created successfully.

Step 2: Import Data

CSS supports importing data to Elasticsearch using Logstash, Kibana, or APIs. Kibana lets you visualize your Elasticsearch data. The following procedure illustrates how to import data to Elasticsearch using Kibana.

1. On the **Clusters** page, locate the target cluster and click **Access Kibana** in the **Operation** column to go to the Kibana login page.

   – Non-security cluster: The Kibana console is displayed.

   – Security cluster: Enter the username and password on the login page and click **Log In** to go to the Kibana console. The default username is **admin** and the password is the one specified during cluster creation.

2. In the navigation pane of Kibana on the left, choose **Dev Tools**.

   The text box on the left is the input box. The triangle icon in the upper right corner of the input box is the command execution button. The text box on the right area is the result output box.

   Figure 1-1 Console page
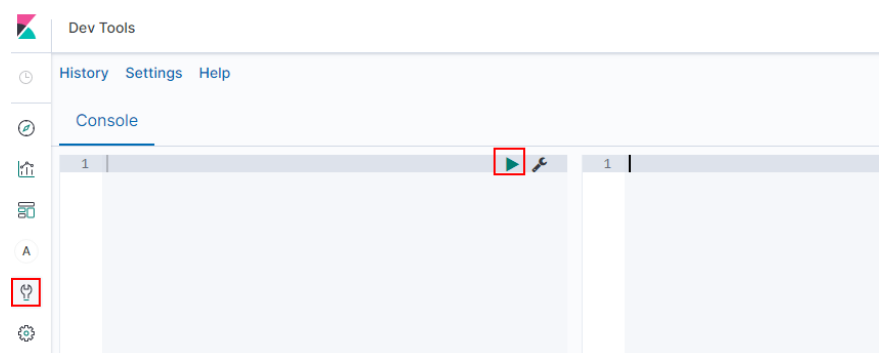
   📖 **NOTE**

   The Kibana UI varies depending on the Kibana version.

3. On the **Console** page, run the following command to create index named **my_store**:

   Versions later than 7.*x*

   ```
   PUT /my_store
   {
     "settings": {
   ```

```
      "number_of_shards": 1
   },
   "mappings": {
       "properties": {
      "productName": {
        "type": "text",
        "analyzer": "ik_smart"
      },
      "size": {
        "type": "keyword"
      }
     }
    }
  }
```

The command output is similar to the following:

```
{
  "acknowledged" : true,
  "shards_acknowledged" : true,
  "index" : "my_store"
}
```

4. On the **Console** page, run the following command to import data to index named **my_store**:

Versions later than 7.*x*

```
POST /my_store/_doc/_bulk
{"index":{}}
{"productName":"Latest art shirts for women in autumn 2017","size":"L"}
{"index":{}}
{"productName":"Latest art shirts for women in autumn 2017","size":"M"}
{"index":{}}
{"productName":"Latest art shirts for women in autumn 2017","size":"S"}
{"index":{}}
{"productName":"Latest jeans for women in spring 2018","size":"M"}
{"index":{}}
{"productName":"Latest jeans for women in spring 2018","size":"S"}
{"index":{}}
{"productName":"Latest casual pants for women in spring 2017","size":"L"}
{"index":{}}
{"productName":"Latest casual pants for women in spring 2017","size":"S"}
```

If the value of the **errors** field in the command output is **false**, the data is imported successfully.

Step 3: Search for Data

- **Full-text search**

  If you access the e-commerce website and want to search for commodities whose names include "spring jeans", enter "spring jeans" to begin your search. The following example shows the command to be executed on Kibana and the command output.

  Command to be executed on Kibana:

  sions later than 7.*x*

  ```
  GET /my_store/_search
  {
    "query": {"match": {
      "productName": "spring jeans"
    }}
  }
  ```

  The command output is similar to the following:

  ```
  {
    "took" : 3,
    "timed_out" : false,
  ```

```
"_shards" : {
  "total" : 1,
  "successful" : 1,
  "skipped" : 0,
  "failed" : 0
},
"hits" : {
  "total" : {
    "value" : 4,
    "relation" : "eq"
  },
  "max_score" : 1.7965372,
  "hits" : [
    {
      "_index" : "my_store",
      "_type" : "_doc",
      "_id" : "9xf6VHIBfClt6SDjw7H5",
      "_score" : 1.7965372,
      "_source" : {
        "productName": "Latest jeans for women in spring 2018",
        "size" : "M"
      }
    },
    {
      "_index" : "my_store",
      "_type" : "_doc",
      "_id" : "-Bf6VHIBfClt6SDjw7H5",
      "_score" : 1.7965372,
      "_source" : {
        "productName": "Latest jeans for women in spring 2018",
        "size" : "S"
      }
    },
    {
      "_index" : "my_store",
      "_type" : "_doc",
      "_id" : "-Rf6VHIBfClt6SDjw7H5",
      "_score" : 0.5945667,
      "_source" : {
        "productName": "Latest casual pants for women in spring 2017",
        "size" : "L"
      }
    },
    {
      "_index" : "my_store",
      "_type" : "_doc",
      "_id" : "-hf6VHIBfClt6SDjw7H5",
      "_score" : 0.5945667,
      "_source" : {
        "productName": "Latest casual pants for women in spring 2017",
        "size" : "S"
      }
    }
  ]
}
}
```

- Elasticsearch supports word segmentation. The preceding command segments "spring jeans" into "spring" and "jeans".

- Elasticsearch supports full-text search. The preceding command searches for the information about all commodities whose names include "spring" or "jeans".

- Unlike traditional databases, Elasticsearch can return results in milliseconds by using inverted indexes.

- Elasticsearch supports sorting by score. In the command output, information about the first two commodities contains both "spring" and

"jeans", while that about the last two products contain only "spring". Therefore, the first two commodities rank prior to the last two due to high keyword match.

- **Aggregation result display**

  The e-commerce website provides the function of displaying aggregation results. For example, it classifies commodities corresponding to "spring" based on the size so that you can collect the number of products of different sizes. The following example shows the command to be executed on Kibana and the command output.

  Command to be executed on Kibana:

  Versions later than 7.*x*

  ```
  GET /my_store/_search
  {
  "query": {
  "match": { "productName": "spring" }
  },
  "size": 0,
  "aggs": {
  "sizes": {
  "terms": { "field": "size" }
  }
  }
  }
  ```

  The command output is similar to the following:

  Versions later than 7.*x*

  ```
  {
    "took" : 3,
    "timed_out" : false,
    "_shards" : {
      "total" : 1,
      "successful" : 1,
      "skipped" : 0,
      "failed" : 0
    },
    "hits" : {
      "total" : {
        "value" : 4,
        "relation" : "eq"
      },
      "max_score" : null,
      "hits" : [ ]
    },
    "aggregations" : {
      "sizes" : {
        "doc_count_error_upper_bound" : 0,
        "sum_other_doc_count" : 0,
        "buckets" : [
          {
            "key" : "S",
            "doc_count" : 2
          },
          {
            "key" : "L",
            "doc_count" : 1
          },
          {
            "key" : "M",
            "doc_count" : 1
          }
        ]
      }
    }
  }
  ```

Step 4: Delete the Cluster

Once you understand the process and method of using Elasticsearch, you can perform the following steps to delete the cluster you created for the example and its data to avoid resource wastage.

> 📖 **NOTE**
>
> After you delete a cluster, its data cannot be restored. Exercise caution when deleting a cluster.

1. Log in to the CSS management console. In the navigation pane on the left, choose **Clusters** > **Elasticsearch**.

2. Locate the row that contains cluster **Sample-ESCluster** and click **More** > **Delete** in the **Operation** column.

3. In the displayed dialog box, enter the name of the cluster to be deleted and click **OK**.